

***Amendments to the Claims***

The listing of claims will replace all prior versions, and listings of claims in the application.

1–10. (*Previously Canceled*).

11. (*Currently Amended*) A microprocessor based system comprising:

a memory controller coupled to a plurality of microprocessors and to a memory shared by the microprocessors;

a plurality of memory caches, each respective memory cache of the plurality of memory caches associated with a respective microprocessor of the plurality of microprocessors;

a global arbiter, coupled to each microprocessor and to said memory controller; and

a plurality of buses:

each respective bus of the plurality of buses coupling a respective microprocessor of the plurality of microprocessors to the global arbiter;

each bus implementing a bus protocol that is different from the bus protocol of at least one other bus of the plurality of buses; and

each bus exhibiting a latency for memory transactions that is different from the latency of at least one other bus of the plurality of buses;

wherein:

the plurality of microprocessors are configured to send a plurality of requests for memory transactions to the global arbiter; and

said global arbiter is configured to:

receive via the plurality of buses the plurality of requests for [[a]] memory transactions from the plurality of microprocessors;

assign a global order to each request of the plurality of requests;

execute according to the global order, a snoop phase for each request of the plurality of requests, wherein each snoop phase comprises determining whether one or

more of the plurality of microprocessors has memory cache data associated with a corresponding request;

wait for all responses to come back in response to all the snoop phases of the plurality of requests; and

respond according to the global order to the plurality of requests after the responses are received for all the snoop phases of the plurality of requests;

whereby said global ordering of the plurality of requests results in data coherence among the plurality of caches, said data coherence being latency independent of the latency differences between the plurality of buses.

12. (Previously Presented) The microprocessor based system of claim 11, further comprising a plurality of bus interfaces;

each respective bus interface of the plurality associated with a respective microprocessor;

each respective bus interface implementing a plurality of bus phases for the memory transactions sent from the plurality of microprocessors to the global arbiter;

wherein said plurality of bus phases implemented by said bus interfaces comprise:

a request phase, where memory requests are submitted via said bus interfaces;

a snoop phase, where a determination is made whether one or more microprocessors of the plurality of microprocessors has data associated with a memory request; and

a response phase, where data associated with a request is received by at least a requesting microprocessor.

13. (Previously Presented) The microprocessor based system as recited in claim 12 wherein said memory requests comprise:

a request to share;

a request to own; and

a request to write back.

14. (Original) The microprocessor based system as recited in claim 13 wherein said request to share results from a read miss in a cache.

15. (Original) The microprocessor based system as recited in claim 13 wherein said request to own results from a write miss in a cache.

16. (Previously Presented) The microprocessor based system as recited in claim 13 wherein said request to write back results from a snoop to a modified cache line.

17. (Previously Presented) The microprocessor based system as recited in claim 11 wherein said determining whether one or more of the plurality of microprocessors has memory cache data associated with a corresponding request comprises communicating a query corresponding to a request to each of the plurality of microprocessors.

18. (Previously Presented) The microprocessor based system as recited in claim 12 wherein a first microprocessor's bus interface couples said first microprocessor to a first bus.

19. (Previously Presented) The microprocessor based system as recited in claim 18 wherein a second microprocessor's bus interface couples said second microprocessor to a second bus.

20. (Previously Canceled)

21. (Original) The microprocessor based system as recited in claim 19 wherein said first bus and said second bus are different.

22. (Original) The microprocessor based system as recited in claim 19 wherein said first bus and said second bus have different timing latencies.

23. (Original) The microprocessor based system as recited in claim 11 wherein said memory controller comprises:

a bus interface for coupling to said global arbiter.

24. (Previously Presented) The microprocessor based system as recited in claim 11 wherein said global arbiter comprises a bus interface for coupling to each of said plurality of microprocessors and said memory controller.

25. (Previously Canceled)

26. (Original) The microprocessor based system as recited in claim 11 wherein said global arbiter comprises:

a plurality of request queues;

a plurality of snoop queues; and

a plurality of response queues;

wherein each of said queues stores a plurality of requests, snoops, and responses, respectively.

27. (Previously Presented) The microprocessor based system as recited in claim 11 wherein said global arbiter comprises ordering and arbitration logic for receiving the plurality of requests from said plurality of microprocessors, and for ordering said plurality of requests into the global order.

28. (Original) The microprocessor based system as recited in claim 27 wherein said ordering and arbitration logic initiates snoops according to said global order.

29. (Previously Presented) The microprocessor based system as recited in claim 28 wherein said snoops have differing latencies depending on the bus protocols of the plurality of buses.

30. – 42. (Previously Canceled)

43. (Currently Amended) A multiphase method ~~protocol~~ for maintaining data coherency between agents that share a common memory through disparate fabrics with different data latencies, ~~the protocol~~ comprising:

presenting in a request phase[[,]] ~~where~~ a plurality of memory requests ~~are presented~~ to a global arbiter, said global arbiter ordering said memory requests into a global order compensating for said differing data latencies, whereby the data coherency is maintained across the agents irrespective of the differing data latencies;

executing in a snoop phase[[,]] ~~during which~~ via the global arbiter ~~executes~~ a snoop corresponding to each memory request of the plurality of memory requests according to said global order, wherein a snoop determines whether one or more agents coupled to the disparate fabrics have data associated with a corresponding memory request; and

providing in a response phase[[,]] ~~where~~ responses to said memory requests ~~are provided~~ according to said global order upon completion of all ~~their associated~~ snoops associated with said memory requests;

whereby said global ordering of the plurality of memory requests results in data coherence among the plurality of agents, said data coherence being latency independent of the latency differences between the disparate fabrics with different data latencies.

44. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 43 wherein the agents comprise a plurality of processor cores.

45. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 44 wherein each of the agents further comprise a cache.

46. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 43 wherein the disparate fabrics comprise a plurality of interfaces.

47. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 46 wherein at least one of said plurality of interfaces comprises a bus.

48. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 43 wherein at least one of said plurality of interfaces comprises a serial fabric.

49. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 48 wherein said serial fabric comprises PCI-Express.

50. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 43 wherein said request phase, said snoop phase, and said response phase pertains to each of said memory requests.

51. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 50 wherein

said memory requests may overlap with said request, snoop, and response phases for another one of said memory requests.

52. (Currently Amended) The multiphase method ~~protocol~~ as recited in claim 43 wherein said memory requests comprise:

memory reads; and  
memory writes.

53. – 58. (Canceled)

59. (Previously Canceled) .

60. (Previously Canceled)

61. (Currently Amended) A computer readable storage medium containing instructions that, when executed by a processor, enable the processor to direct:

- a global arbiter, coupled to each of a plurality of processor cores;

- a plurality of buses:

- each respective bus of the plurality of buses coupling a respective microprocessor of the plurality of processor cores to the global arbiter;

- each bus implementing a bus protocol which is different from the bus protocol of at least one other bus of the plurality of buses; and

- each differing bus exhibiting a latency for data transactions which is different from the latency of at least one other bus of the plurality of buses;

- a bus interface for each of the processor cores to couple the processor cores to the global arbiter via the plurality of buses; and

- a bus interface to couple the global arbiter to a memory;

wherein said instructions comprise program code means for causing the processor to direct the global arbiter to:

- receive a plurality of memory requests from the plurality of processor cores;

- establish a global order for the requests; and

- initiate a snoop phase for each request of the plurality of requests according to the global order, each snoop phase comprising the global arbiter querying the plurality of processor cores to determine whether they contain data pertaining to the plurality of memory requests; and

- wherein:

- the processor cores respond to the snoop phase initiated by the global arbiter at different times; and

- said global ordering of the memory requests results in data coherence among the data contained by each processor core, said data coherence being latency independent of the latency difference between the plurality of buses.

62. (Previously Presented) The computer readable storage medium as recited in claim 61 wherein said instructions enable the processor to direct the global arbiter to implement:

request logic, for receiving requests from the processor cores;  
snoop logic, for communicating queries associated with received requests from the global arbiter to the processor cores; and  
ordering logic, for establishing a global order for received requests, and for causing said snoop logic to communicate the received requests to the processor cores according to the global order.

63. (Currently Amended) The computer readable storage medium as recited in claim 62 wherein said instructions enable the processor to direct the global arbiter to implement:

response logic, for causing responses to the received requests to be provided upon completion of all of ~~all~~ the snoop requests.

64. (Previously Presented) The computer readable storage medium as recited in claim 62 wherein said instructions enable the processor to direct the global arbiter to implement:

response logic, for causing responses to the received requests to be provided to the processor cores according to the global order.

65– 69. (Previously Canceled)

70. (Currently Amended) The multiphase method ~~protocol~~ of claim 43, wherein a snoop further comprises the global arbiter communicating a query corresponding to a memory request from a first agent to at least an agent of the agents that share the memory.



71. (Currently Amended) The multiphase method ~~protocol~~ of claim 43, wherein ~~[[in]]~~ determining in the snoop phase whether one or more agents coupled to the disparate fabrics have data associated with a given memory request comprises~~[[,]]~~ ~~the global arbiter is further configured to:~~

communicating from the global arbiter ~~communicate~~ a query corresponding to the given memory request ~~[[from]]~~ of a first agent to all agents that share the memory; and

waiting until a response to the query is received at the global arbiter from each agent before responding to the given memory request.

72. (Previously Presented) A memory controller coupled to a memory and to a plurality of agents configured to share the memory;

said memory and said plurality of agents coupled via a plurality of buses, each respective agent having a respective bus, each bus of the plurality of buses implementing a bus protocol with a different data latency from the data latency of at least one other bus;

wherein the memory controller is configured to:

receive a plurality of memory requests;

assign a global order to each of the plurality of memory requests, said global order compensating for the differences in data latencies between the plurality of buses;

execute according to the global order, a snoop corresponding to each request, wherein each snoop comprises determining whether one or more of the plurality of agents has data associated with a corresponding request;

wait for all responses to come back in response to all the snoop phases of the plurality of requests; and

respond to each request according to the global order;

whereby a data coherency is maintained across the agents irrespective of the differences in data latencies between the plurality of buses.

73. (Previously Canceled).

74. (Previously Presented) The memory controller of claim 72, wherein in determining whether another an agent of the plurality of agents has data associated with a first request from a first agent, the memory controller is further configured to communicate a query corresponding to the first request to at least an agent of the plurality of agents.

75. (Previously Presented) The memory controller of claim 72, wherein in determining whether one or more of the plurality of agents has data associated with a given request, the memory controller is further configured to:

communicate a query corresponding to the given request to at least an agent of the plurality of agents; and

wait until a response to the query is received from the at least an agent before responding to the given request.

76. (Previously Presented) The memory controller of claim 72, wherein the plurality of agents comprise at least one of a microprocessor cores and I/O devices.

77. (Previously Presented) The memory controller of claim 76, wherein each of said microprocessor cores includes a cache.

78-80. (Previously Canceled)

81. (Previously Presented) The memory controller of claim 72, wherein each agent monitors requests to determine whether it holds data associated with a request from one of the plurality of agents.

82. (Previously Presented) The memory controller of claim 72, wherein if a first agent determines that another one of the plurality of agents requests data that is within the first agent, the first agent informs the memory controller.

83. (Currently Amended) The memory controller of claim 72, further comprising:  
request logic, for receiving the memory requests; and  
snoop logic, for initiating said snoops to the plurality of agents; ~~and~~  
wherein said initiated snoops are latency independent with respect to said  
requests.

84. (Previously Presented) The memory controller as recited in claim 83, wherein  
said request logic comprises a plurality of queues for receiving said requests from the  
plurality of agents and for presenting said requests to said ordering logic.

85. (Previously Presented) The memory controller as recited in claim 83, wherein  
said snoop logic comprises a plurality of queues for storing snoops conforming to said  
global order.

86. (Previously Presented) The memory controller as recited in claim 83 wherein  
said ordering logic causes said snoop logic to initiate snoops according to said global  
order.

87. (Previously Presented) The memory controller as recited in claim 83 wherein  
said global order is established according to a first-in first-out rule.

88. (Previously Presented) The memory controller as recited in claim 83 wherein  
said ordering logic insures that responses to said requests are performed according to  
said global order.

89. (Previously Presented) The memory controller as recited in claim 83 further  
comprising:

response logic for ensuring that responses to said requests are performed  
according to said global order.

90-91. (Previously Canceled)

92. (Previously Presented) The memory controller as recited in claim 72 wherein at least one of said plurality of buses supports a plurality of virtual channels.

93. (Previously Presented) The memory controller as recited in claim 92 wherein said at least one of said plurality of bus interfaces comprises a bus interface to PCI-Express.